

Hybrid model for software development: an integral comparison of DevOps automation tools

Poonam Narang, Pooja Mittal

Department of Computer Science and Applications, Maharshi Dayanand University (MDU), Haryana, India

Article Info

Article history:

Received Mar 23, 2022

Revised May 21, 2022

Accepted May 31, 2022

Keywords:

Automation

Automation tools

DevOps

Software development life cycle

Software development

ABSTRACT

DevOps is a fine fusion of development and operations teams together to deliver more efficient, reliable and quality software. DevOps make use of alternative set of automation tools for different development stages of software. The research presented in this paper analyses selective automation tools to provide comprehensive and comparative tabular analysis followed by graphical comparison according to latest Google Trends. Best performer tools out of these alternative tool sets are grouped together into integrated tool chain (ITC) in order to escalate DevOps performance in future. A hybrid automated model for software development using these selective automation tools from the ITC is also proposed. This analytic comparison will prove to be a big utility for young researchers, students and software developers to be cognizant with DevOps automation culture. Study of other tools or enhancement of the proposed hybrid model could also be considered as a part of future research.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Poonam Narang

Department of Computer Science and Applications, Maharshi Dayanand University (MDU)

Delhi rd, near Delhi Bypass, Rohtak, Haryana 124001, India

Email: poonam.mehta20@gmail.com

1. INTRODUCTION

Successful and speedy software releases depend ponderously on effective communications in-between development and operations teams together. Earlier these teams worked in silos under traditional or Agile software development models or methodologies. DevOps, on the other hand, brings Dev (development) and Ops (operations) teams close to each other with the use and support of alternative set of automation tools available for different stages in software development. Different tools employed by DevOps include confluence, GitHub, Bitbucket, Jenkins, Chef, Puppet, Docker, and Splunk to ease out the tasks of these teams. DevOps tools work as both software as a service (Saas) and a platform as a service (PaaS) for both teams of development and operations.

The work under this analytic comparative research includes in-depth study of different policies or automation tools employed in DevOps culture. These sets of alternative tools impart automation to all stages of software development. This paper also suggests the best tool available at different software development stages based on their comprehensive and comparative analysis from alternative tool sets. These best chosen automation tools, termed as selective tool chains, work in collaboration with the aim of accelerating or speeding up the software development and delivery process up to a much greater extent. A hybrid automated model comprising these selective automation tools from DevOps alternative set of tools is also presented. This extensive or evaluative analysis will be useful for different software developers or young researchers and students to understand DevOps modus operandi and culture. Current research will also be beneficial for DevOps adoption and usage of surrogate automation tools during different software development stages.

As of future scope, DevOps automation tools can be studied and analysed to be selected as different tool chains depending upon the project requirements. These tool chains will be of greater comfort or support for software developers to deliver software in smaller sprint times along with the fulfilment of high quality and efficiency requirements. Using these different tool chains another hybrid model can also be proposed as part of further work.

The remainder of this paper is structured as shown in the following schematic diagram or flowchart Figure 1 depicts the analytic and comprehensive comparisons sequence of alternative set of automations tools inked at different stages of software development included in the current work. Present work also proposes integrated tool chain (ITC) in the form of hybrid model for automation as the last step shown in flow diagram.

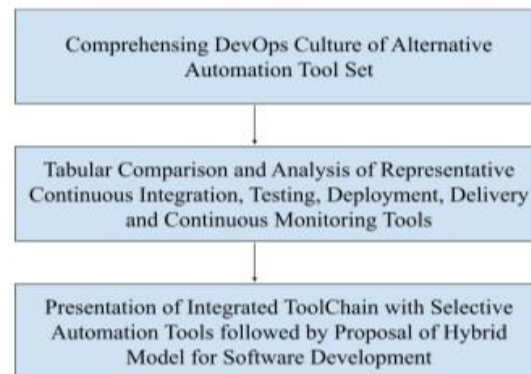


Figure 1. Schematic flow diagram for the work covered in the present research paper

2. RELATED WORK

Automation tools employed in DevOps culture for software development till delivery, deployment and maintenance are reviewed from different literature already existing in terms of research papers. Bobrov *et al.* [1] consider DevOps as the first competency along with Agile that is required by the industry. Authors in this paper conduct experiments on education and industry level to prepare engineers in the best possible way. This paper studies different principles, practices, tools and architectures to throw light on many aspects of DevOps approaches. Another IEEE Paper on DevOps [2] talks about the automation of DevOps processes using cloud and non-cloud DevOps automation tools. Target of their paper is to move DevOps to the cloud and make it more agile at software development and operations. The primary aim of their research is to extend DevOps processes and automation into public or private. Leite *et al.* [3] in their survey paper on DevOps challenges and concepts investigates and discusses very well about DevOps challenges from the perspective of engineers, managers and researchers. Their survey reviews the literature and also develops a conceptual map that correlates DevOps with its automation tools. Some of the practical implementation along with challenges also discussed with critical exploration.

In another research on DevOps by Mishra and Otaiwi [4] discusses software quality in parallel and aims at analyzing the implications of DevOps features on software quality. This study presents systematic mapping of DevOps impact on software quality. The research was mainly focussed on automation, culture, continuous delivery and fast feedback of DevOps. Olguin [5] and Jabbari *et al* [6] in their papers highlight that DevOps acts as a movement to automate the tasks of continuous delivery of new software updates while at the same time guaranteeing their correctness and reliability. Jabbari *et al* [6] also conducted systematic literature review on the definition of DevOps and agrees that DevOps extends the agility component in software development paradigm.

Hussaini in DevOps paper [7] accepts the emerging of DevOps paradigm as a response to the growing knowledge of existing gap of 4 Cs (Communications, Cooperation, Culture and Collaboration) between development and operation teams functions of an organisation. Authors also accept “Wall of Confusion” between these teams. This “Wall” is caused by combination of conflicting motivations among people, processes and technology/tooling. Hence, need for strengthening harmonization of Dev and Ops teams arises. The model was also outlined in [7] for enhancing effectiveness and efficiency of DevOps stakeholder’s interest.

The literature survey studied, specified or discussed so far lacks in individual automation tool discussion that was taken out as the primary objective of current research work. We also analyzed

comparatively about alternative set of automation tools available at different software development stages in DevOps followed by presentation and proposal of ITC along with hybrid model for software development.

3. DEVOPS-EMPLOYING AUTOMATION AT DIFFERENT SOFTWARE DEVELOPMENT STAGES

DevOps, culture of automation tools relies heavily on 5 Cs of software development. These are: i) continuous integration, ii) continuous testing, iii) continuous delivery, iv) continuous deployment, and v) continuous monitoring. the tasks of these Cs are distributed among dev and ops teams as depicted in the following diagram. Figure 2 clearly depicts the progression order for different phases of plan, code, build, test under development team and release, deploy, operate, Monitor under operations team. A much bigger list of automation tools are available for different software development and operations stages for which DevOps employs an alternative set of tools separately. Some of these tools are being selected as representatives for comparisons in different C's of DevOps based on their popularity and commonly usage along with availability as open-source software. Other than these properties, automation tools shortlisted for deeper comparison also have many other unique features like compatibility and container ability with other tools, good web interface, easy installation, scalability, better understandability, less need of other resources, parallel execution of test cases, good support of different programming languages, and vulnerability. These representative automation tool sets are summarised in the Table 1.

Table 1 of automation tools, mentioned under different DevOps stages, will be compared comprehensively and analytically throughout this research work. This analytic comparison will move towards the introduction of integrated tool chain for specific software applications. Sections covers the analytical and comprehensive review of these alternative and most commonly followed automation tools.

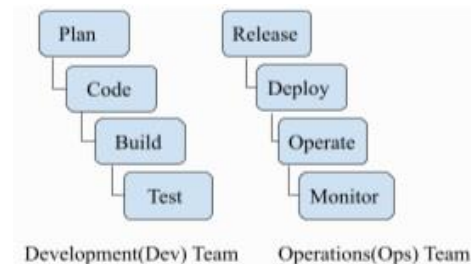


Figure 2. schematic progression order of different stages of software development employed by Dev and Ops teams in DevOps

Table 1. Alternative set of DevOps automation tools at different software development stages

Development stage	Tool set
Continuous integration	Jenkins, Teamcity, Buildbot
Continuous testing	Selenium, JMeter, TestComplete
Continuous delivery	GitHub Actions, Azure DevOps, GitLab
Continuous deployment	Puppet, Ansible, Chef
Continuous monitoring	Splunk, Nagios, Sensu

3.1. Continuous integration tools

Continuous Integration is a practice that helps developers deliver software in a much more reliable and predictable manner. Earlier traditional integration was observed in which each developer gets its code copy from the central repository. All the developers begin with the same starting point and work on adding new features or fixing bugs and making progress by working either in a team or on their own. Since all the developers are working simultaneously on the same code and everyone is trying to add new features or fixing earlier existing bugs, thus complicating the task of the central repository to maintain a single and updated copy of the code. These problems of maintaining a stable version of the code gives rise to the need of continuous integration.

Continuous integration restricts the developers to integrate their updated code with the central repository on a daily basis or for small time periods so that updated, current and more stable version of code is available at all time with no conflicts. To accomplish these, DevOps adopts automation tools for

continuous integration from many alternatives viz. Jenkins, Bamboo, Buddy, Circle CI, Teamcity, Codeship, and Cruise control. From all these available alternatives, the current review considers Jenkins, Teamcity and Buildbot as the representative tools for continuous integration. Based on existing literature review or study, these representative tools are compared with one another in consideration to different parametric or performance evaluators as shown in the Table 2.

Table 2. Comparison of representative continuous integration tools wrt different parameters [8]-[13]

Parameter under consideration	Jenkins	Teamcity	Buildbot
Ecosystem of plugins	huge set of plugins exist	not a big list of plugins	big range of plugins exist
Hosting (on-premise/self- hosted)	Both	only on-premise	only on- premise
Integration with other tools	Yes [8]	Only with popular cloud solutions	Yes but with limited tools
Ease of use	Not as much user friendly	very much user friendly	Easier to use and customize
Open Source	Yes	for first 100 builds	Yes

Table 2 shown of comparison depending on different parametric or performance evaluators, Jenkins wins the feature race among but it lacks in ease of use and customization features. Jenkins is fewer users friendly in contrast to other tools but still Jenkins can be concluded as the most visual and commonly followed continuous integration tool when it comes to choosing functionality as the important need. The Google trends report of past 12 months for Jenkins, Teamcity and Buildbot, which also indicated that Jenkins is much popular than other continuous integration tools. Trend in Figure 3 clearly depicts the increasing use of Jenkins tool over the year in comparison with TeamCity and Buildbot tools. Thus, by seeing Google trends report, it can easily be said that Jenkins is the most observed tool but depending on the requirements other tools can also be preferred over another.

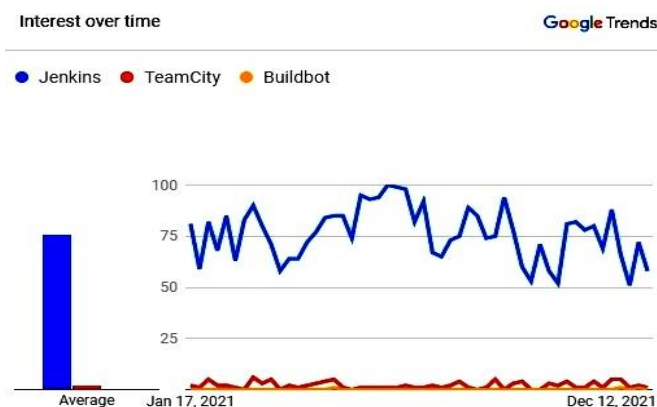


Figure 3. Jenkins vs TeamCity vs Buildbot-interest over time by Google trends [14]

3.2. Continuous testing tools

Continuous testing is concerned with the scheduling of automation tests after every feature update performed by the developers. On the other hand, the traditional way of software testing was hand-off centric which included handing off the software from one team to another. A software project would have definite development and deploy phases. Each team wanted more time to ensure quality and best deployment whereas business or organisation wants faster delivery. Here arises the requirement of a new testing methodology that evolves in the form of continuous testing.

Continuous testing generally implies the testing performed in an uninterrupted manner and also on a continuous basis. DevOps employs automation tools to achieve continuity in testing. Many alternative tools that are available for continuous testing are QuerySerge, Travis, Selenium, JMeter, TestComplete, Appium, Watir, and Eggplant. Existence of many alternative testing automation tools makes it utmost important to select appropriate tools required for the current application. The current comparative analysis considers Selenium, Jmeter, and TestComplete as representatives for continuous testing. These representatives are studied through existing literature review and compared on the basis of performance evaluators or parameters. Comparative analysis of these tools are shown in the Table 3.

Table 3. Comparison of representative continuous testing tools with respect to different performance evaluators [15]-[19]

Parameter under consideration	Selenium	Jmeter	TestComplete
Development Platform Supported	Window, Linux, Mac	cross- platform	Windows
IDE Support	support IDE	support extended IDE [17]	only support in-built IDE
Execution time taken	much time taken to write scripts	Quick	takes less execution time
Cost	Open Source/Free Tool	Open Source	Highly paid
Ease in tool adoption	need expertise	very easy to learn	easy to adapt[19]

Based on the results of Table 3, it can be concluded that Selenium is the most commonly observed automation tool for continuous testing as it is free of cost but this tool lacks in easy learning. Selenium needs expertise to write scripts. Figure 4 shows the popularity/usage trend in the past 12 months for considered continuous testing tools. Google trends of Figure 4 also reveal the increased demand of selenium over the period of last 12 months as compared to other alternative testing tools. But other testing automation tools are also followed for different applications depending on project specifications.

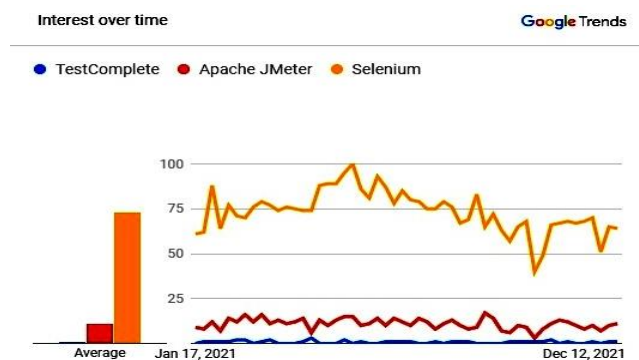


Figure 4. Interest over time for Selenium, TestComplete and Jmeter continuous testing tools [20]

3.3. Continuous delivery tools

Continuous delivery in DevOps software development culture is a phase in which code changes are built automatically, tested and also prepared for release or production. It is a practice that enables developers to build, test and release code changes to production teams in an incremental manner. Earlier software delivery, on the other hand, includes more cost, time and risk of delivery changes that is reduced to a greater extent by introducing continuity in delivery through incremental code changes and releases. Different alternative automation tools exist for continuous delivery viz. Jenkins, GitLab, Travis, Ansible, CircleCI, XLDeploy, AzureDevOps, Harness, and GitHub Actions. Current research or review takes GitLab, Azure DevOps, and GitHub Actions as representatives for continuous delivery. A comparative analysis of these selected tools on the basis of parametric evaluators is shown in Table 4. On the basis of evaluative analysis in terms of Table 4, it can be concluded that GitLab is the best tool among all others because of its ease of understanding and usage along with requirement of less scripting knowledge. Stack overflow trend in Figure 5 also confirms the same about the tool.

Figure 5 shows clearly that GitLab tops among GitHub actions and other delivery tools as per the trend report given by stack overflow. Also GitLab opt the feature of continuous verification in software delivery that escalates the release quality up to much bigger extent. But still depending upon the requirements on a specific project, other tools are also preferred.

Table 4. Comparative analysis of representative continuous delivery tools based on different parameters [21]-[25]

Parameter	GitLab	GitHub actions	Azure DevOps
Continuous verification	Yes	No	Yes
Scripting requirements	No	No	No
SaaS and On- Premises services	Yes	Yes but with limited features in On-Premises	Yes
Quality of product	Best	Good	Good
Leaning of tool	Easy	Moderate	Moderate

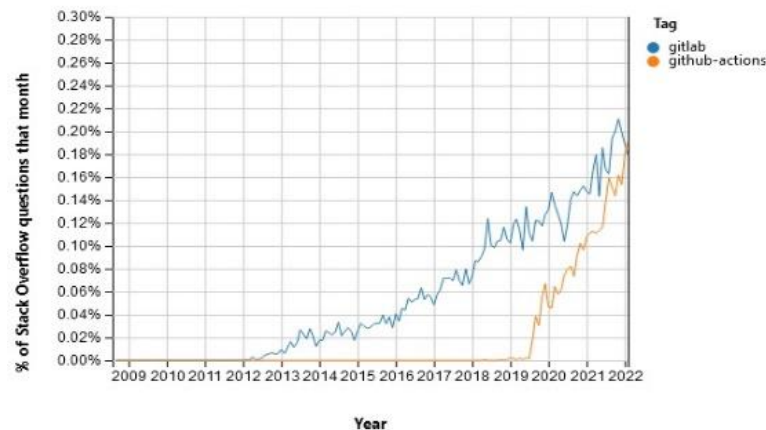


Figure 5. Interest over time demand trend according to stack overflow over past around 8 years [26]

3.4. Continuous deployment tools

Continuous deployment is a software release process that is used for immediate autonomous deployment to the production environment after automated test validation. Continuous deployment offers remarkable benefits to modern software businesses. It also allows businesses to respond to teams along with meeting changing and increasing market demands to deploy and validate new features rapidly [9].

Continuous deployment differs from continuous delivery in the context of human intervention later before deploying the automated tested release to the production environment. Different automation tools for continuous deployment are Ansible, Octopus\Deploy, Docker, Puppet, Chef, and DeployBot. Present research has taken Puppet, Ansible, and Chef as continuous deployment representatives because of their most common usage in development industries. These automation tools are then compared with one another on the basis of different performance evaluators. A comparison is shown in Table 5.

Based on Table 5 comparative evaluation, ansible can be concluded as the best continuous deployment automation tool because of its ease of installation, usage, quality and compatibility with other tools. Following report of Coralogix under stack overflow trends clearly depicts the increasing usage of Ansible by software professionals over the last 7-8 years. Ansible, as shown in Figure 6 of trend graph, leads the marathon among other deployment tools like Terraform, Chef and Puppet. Rest tools can also be taken into consideration and are used widely depending on the particular project.

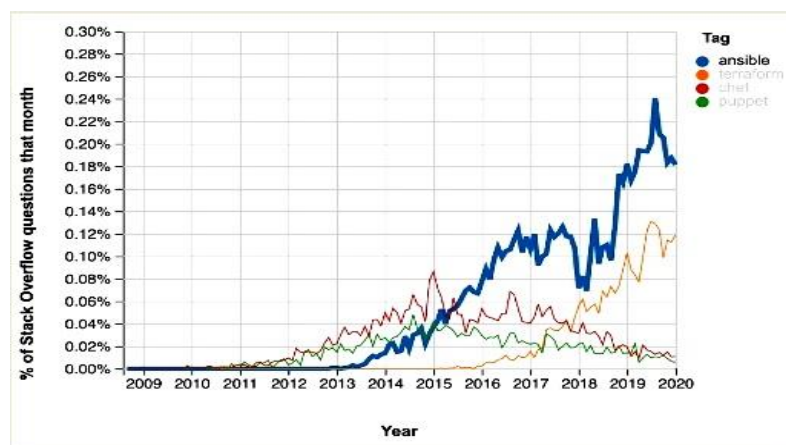


Figure 6. Continuous deployment tools comparison according to stack overflow trends [32]

3.5. Continuous monitoring tools

Continuous Monitoring takes place at the end of DevOps lifecycle phases of software development. After the software is released into the production environment, continuous monitoring comes into action to notify development and operations teams about any kind of issues arising in production. It keeps an eye on

correct working of the software and provides feedback if anything goes wrong so as to fix it at its earliest. Radically, continuous monitoring can be defined as an automated process through which DevOps teams can be in compliance with any kind of issues or threats that are observed or detected at any stage of DevOps life cycle.

Continuous Monitoring enables faster and better response to changing needs of customers in contrast to traditional monitoring methods. DevOps makes use of automation tools to achieve the targets of continuous monitoring. Several continuous monitoring tools that exist are Tenable, Whatsup, Sensu, Insight, Nagios, SolarWinds, and Splunk. This paper takes Nagios, Sensu and Splunk to compare analytically to choose best among all. Following Table 6 shows parameters based evaluative comparison of these tools.

On the basis of comparative analysis Table 6, it can be concluded that Nagios performance overcomes from the other continuous monitoring automation tools in terms of its ease of usage and for freeware availability. Splunk, on the other hand, is also preferred by larger companies as it is very expensive but user friendly at the same time. As depicted in Figure 7, we can observe that Nagios still leaves other continuous monitoring automation tools behind when it comes to usability trend. Other automation tools can also be taken into consideration or observed depending on the continuous monitoring requirements.

Table 6. Comparative analysis based table for different continuous monitoring tools based on evaluative parameters [33]-[36]

Parameter under consideration	Nagios	Sensu	Splunk
Dashboard features	Better	Good	Best
Documentation required	Yes	Yes	Less required
Ease of use	Very easy	Moderate level	Easy
Alert mechanism	Best	Good	Better
Availability cost	Free	Basic version free	Expensive

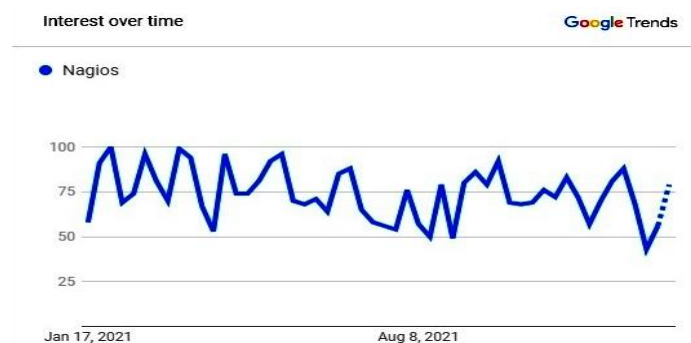


Figure 7. Interest over time for Nagios monitoring tool according to Google trends for the past 12 months [37]

4. PROPOSED HYBRID AUTOMATION MODEL

In today's complex business world, industries either by their own choice or by situational, are choosing a hybrid model solution proposed by IT industries. Non-IT Enterprises are best served by implementing a “best of breed” solution that means combination of main domain specific SaaS and PaaS solutions. Now, it becomes a challenge to make product delivery work seamlessly around these characteristics.

These challenges of delivery work along with achieving their characteristics can be possible with the introduction of partial or complete automation in software delivery, operations as well as in management processes. This paper discusses DevOps culture in IT and even in non-IT industries that provides automation in development and operations using different tools, and attempts to solve these issues. The underlying performance evaluative comparison of different automation tools accelerates towards the design of an ITC of these selective representative tools. This tool chain selection optimizes the performance of the delivery life cycle by removing different impediments at each stage. The ITC designed in turn leads towards the evolution of a hybrid model of automation tools for software development. Following diagram clearly depicts the proposed hybrid automated model consisting of these selective tool chain at different stages of DevOps culture.

In Figure 8 of proposed hybrid model, text outside the ovals of DevOps phases indicates the representative automation tools for different parametric comparison and text inside the diagram represents the highest performer tool for the phase. These best or highest performer tools form the ITC for DevOps phases and the hybrid model of automated tools proposes the inclusion of selected tool chain for the project in order to achieve automated software process in development, deployment till delivery and maintenance. The usage or evidence of ITC can also be observed from the following diagram by their real time curves of high demand.

Figure 9 displays the Google trends for usability of automation tools included in ITC. Thus it can be said as per trends report that proposed model will improve the software release time up to a much greater extent with the cutting of automation tool selection time either before or after the completion of every DevOps phases. Software developers will now be able to start the automation process from the development phase that will be carried automatically till deployment and maintenance with the support of hybrid model proposed toolset. In other words, the proposed model will escalate the delivery and deployment process and thereby enhance the quality much more efficiently. In fact, the proposed ITC in hybrid model will serve as a big advantage to both IT and non-IT industries.

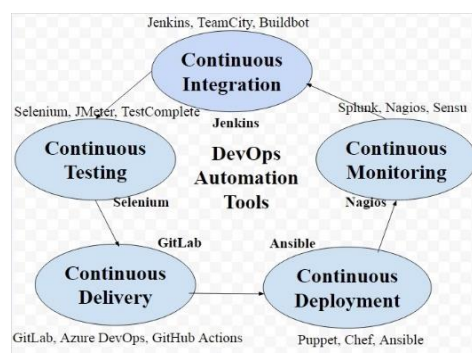


Figure 8. Proposed hybrid automated model of selective tool chain from DevOps alternative automation tools set

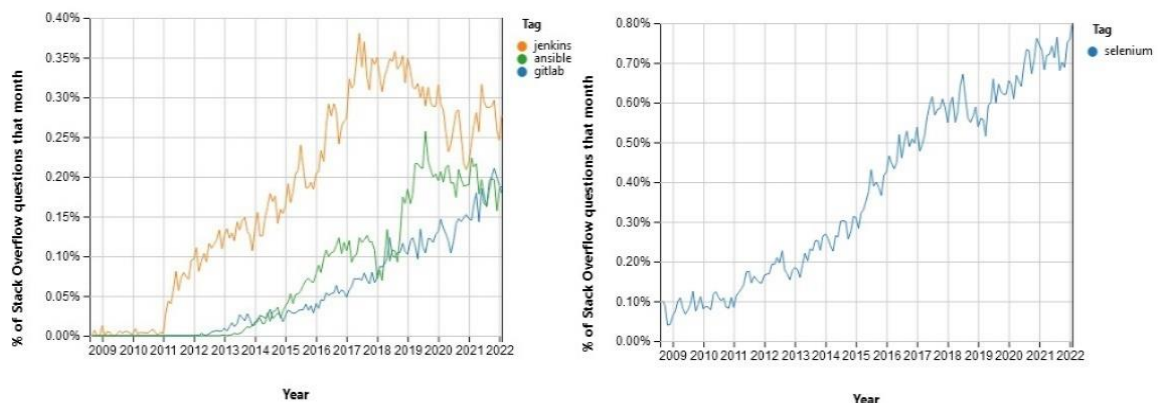


Figure 9. Proposed ITC interest over time as per Google trends [38], [39]

5. CONCLUSION AND DISCUSSION

As we have seen, how hand to hand coordination between both development and operations teams can escalate the software release or delivery process up to a much greater extent. DevOps bridges the gap between these two teams by using the correct automation tool set at different software development stages. DevOps has been an emerging topic for many years, but it's still very common for many enterprises or IT industries to feel overwhelmed by the complex structure of DevOps automation tools and used to getting hung up on which tool to prefer at which stage. This paper studies and analytically compares these alternative automation tools employed at different software development stages to design DevOps own ITC, to elevate DevOps performance and targets to get solutions to these development industry challenges.

The work carried out here also suggests designing a hybrid automated model of these selective tool chain from the set of alternative automation tools in DevOps culture. This proposed hybrid automated model will not only improvise the efficiency of the delivered software but also reduce the release time drastically. This performed comprehensive and comparative evaluation of automation tools through underlying research will be useful for the young researchers/ students to get an in-depth view of DevOps culture. Also an analysis or the impact of using and design of other selective tool chains may be carried out as a part of future research. This study and analysis can also be used to design another hybrid model by synthesizing different tool chains from the selective remaining automation tools as per the requirements of the particular system under consideration.




REFERENCES

- [1] E. Bobrov, *et al.*, "DevOps and its philosophy: education matters!", *Microservices*, Cham: Springer, 2020.
- [2] P. Agrawal and N. Rawat, "Devops, A new approach to cloud development & testing," *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, 2019, pp. 1-4, doi: 10.1109/ICICT46931.2019.8977662.
- [3] L. Leite, C. Rocha, F. Kon, D. Milojevic and P. Meirelles, "A survey of DevOps concepts and challenges," *ACM Computing Surveys*, vol. 52, no. 6, p. 127, 2020, doi: 10.1145/3359981.
- [4] A. Mishra and Z. Otaiwi, "DevOps and software quality: a systematic mapping," *Review Article in Computer Science Review*, vol. 38, p. 100308, 2020, doi: 10.1016/j.cosrev.2020.100308.
- [5] R. Olguin, "DevOps challenges and implications," 2019.
- [6] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, "What is DevOps? a systematic mapping study on definitions and practices," *XP '16 Workshops: Proceedings of the Scientific Workshop Proceedings of XP2016*, 2016, p.12, 10.1145/2962695.2962707.
- [7] S. W. Hussaini, "Strengthening harmonization of development (Dev) and operations (Ops) silos in IT environment through systems approach," *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 178-183, doi: 10.1109/ITSC.2014.6957687.
- [8] A. M. Berg, *Jenkins continuous integration cookbook: Over 90 recipes to produce great results from Jenkins using pro-level practices, techniques, and solutions*, Birmingham, UK: Packt Publishing, 2015
- [9] A. Schaefer, M. Reichenbach, and D. Fey, "Continuous integration and automation for DevOps," in *IAENG Transactions on Engineering Technologies. Lecture Notes in Electrical Engineering*, Dordrecht: Springer, 2012.
- [10] J. Bosch, "Continuous software engineering: An Introduction," *Continuous Software Engineering*, pp. 3-13, 2014, doi: 10.1007/978-3-319-11283-1_1.
- [11] J. Wettinger, V. Andrikopoulos and F. Leymann, "Automated capturing and systematic usage of DevOps knowledge for cloud applications," *2015 IEEE International Conference on Cloud Engineering*, 2015, pp. 60-65, doi: 10.1109/IC2E.2015.23.
- [12] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery," *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, 2015, pp. 78-82, doi: 10.1109/INTECH.2015.7173368.
- [13] J. Smeds, K. Nybom, and I. Porres, "Devops: a definition and perceived adoption impediments," in *International Conference on Agile Processes in Software Engineering, and Extreme Programming*, 2015, pp. 166-177, doi: 10.1007/978-3-319-18612-2_14.
- [14] Google Trends Compare (Jenkins, Teamcity, Bamboo), [Online]. Available: <https://trends.google.com/trends/explore?geo=IN&q=%2Fm%2F0g9x0gr%2Fm%2F026whd1%2Fm%2F0db45t> (accessed Mar. 22, 2022).
- [15] B. Chen, "Improving the Software Logging Practices in DevOps," *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2019, pp. 194-197, doi: 10.1109/ICSE-Companion.2019.00080.
- [16] S. M. Mohammad, "Improve software quality through practicing DevOps," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 6, Issue 1, pp. 251-256, 2018,
- [17] M. Herring, "Continuous everything in DevOps," Accenture. <https://www.accenture.com/us-en/blogs/software-engineering-blog/herring-continuous-everything-in-devops> (accessed Mar 22, 2022).
- [18] S. Elbaum, G. Rothermel, and J. Penix, "Techniques for improving regression testing in continuous integration development environments," in *FSE 2014: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014 pp. 235-245, doi: 10.1145/2635868.2635910.
- [19] E. Engström and K. Petersen, "Mapping software testing practice with software testing research - SERP-test taxonomy," *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2015, pp. 1-4, doi: 10.1109/ICSTW.2015.7107470.
- [20] Google Trends Compare (TestComplete, Apache JMeter, Selenium), [Online]. Available: <https://trends.google.com/trends/explore?geo=IN&q=%2Fm%2F04yfp7b,%2Fm%2F04ypch,%2Fm%2F0c828v> (accessed Mar. 22, 2022).
- [21] D. Farley and J. Humble, *Continuous delivery: reliable software releases through build test and deployment automation*, New York: Addison-Wesley, 2010.
- [22] L. Chen, "Towards architecting for continuous delivery," in *2015 12th Working IEEE/IFIP Conference on Software Architecture*, 2015, pp. 131-134, doi: 10.1109/WICSA.2015.23.
- [23] J. Humble and J. Molesky, "Why enterprises must adopt devops to enable continuous delivery," *Cutter IT Journal*, vol. 24, no. 8, pp. 6-12, 2011.
- [24] S. Krusche and L. Alperowitz, "Introduction of continuous delivery in multi-customer project courses", in *ICSE Companion 2014: Companion Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 335-343, doi: 10.1145/2591062.2591163.
- [25] L. Chen, "Microservices: architecting for continuous delivery and DevOps," in *2018 IEEE International Conference on Software Architecture (ICSA)*, 2018, pp. 39-397, doi: 10.1109/ICSA.2018.00013.
- [26] Stack Overflow Trends, Tags (Azure- DevOps, GitLab, Github-Actions), [Online]. Available: <https://insights.stackoverflow.com/trends?tags=azure-devops%2Cgitlab%2Cgithub-actions> (accessed Mar 22, 2022).
- [27] J. Wettinger, U. Breitenbucher, and F. Leymann, "Compensation and convergence - comparing and combining deployment automation approaches," *International Journal of Cooperative Information Systems*, vol. 24, no. 3, p. 1541001, 2015, doi: 10.1142/S0218843015410014.
- [28] T. Savor, M. Douglas, M. Gentili, L. Williams, K. Beck and M. Stumm, "Continuous deployment at Facebook and OANDA," in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 2016, pp. 21-30.




- [29] D. Ståhl and J. Bosch, "Modeling continuous integration practice differences in industry software development," *Journal of Systems and Software*, vol. 87, pp. 48-59, 2014, doi: 10.1016/j.jss.2013.08.032.
- [30] C. A. Cois, J. Yankel and A. Connell, "Modern DevOps: optimizing software development through effective system interactions," in *2014 IEEE International Professional Communication Conference (IPCC)*, 2014, pp. 1-7, doi: 10.1109/IPCC.2014.7020388.
- [31] M. D. Jong and A. V. Deursen, "Continuous Deployment and schema evolution in SQL databases," *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, 2015, pp. 16-19, doi: 10.1109/RELENG.2015.14.
- [32] "Stack Overflow trends." <https://insights.stackoverflow.com/trends?tags=ansible%2Cterraform%2Cpuppet%2Cchef-infra> (accessed Mar 22, 2022).
- [33] M. A. L. Peña, J. Díaz, J. E. Pérez and H. Humanes, "DevOps for IoT systems: fast and continuous monitoring feedback of system availability," in *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10695-10707, Oct. 2020, doi: 10.1109/IIOT.2020.3012763.
- [34] M. Rajkumar, A. K. Pole, V. S. Adige and P. Mahanta, "DevOps culture and its impact on cloud delivery and software development," *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring)*, 2016, pp. 1-6, doi: 10.1109/ICACCA.2016.7578902.
- [35] D. Stahl, T. Martensson and J. Bosch, "Continuous practices and devops: beyond the buzz, what does it all mean?," *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2017, pp. 440-448, doi: 10.1109/SEAA.2017.8114695.
- [36] A. Schaefer, M. Reichenbach, and D. Fey, "Continuous integration and automation for DevOps," in *IAENG Transactions on Engineering Technologies. Lecture Notes in Electrical Engineering*, Dordrecht: Springer, 2013.
- [37] Google Trends Explore (Nagios). [Online]. Available: <https://trends.google.com/trends/explore?q=%2Fm%2F03hm4g&hl=en-US> (accessed Mar 22, 2022).
- [38] Stack Overflow trends. [Online]. Available: <https://insights.stackoverflow.com/trends?tags=jenkins%2Cazure-devops%2Cansible> (accessed Mar 22, 2022).
- [39] Stack Overflow trends. [Online]. Available: <https://insights.stackoverflow.com/trends?tags=selenium> (accessed Mar 22, 2022).

BIOGRAPHIES OF AUTHORS



Poonam Narang    Research Scholar, pursuing PhD from the Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, Haryana under the supervision of Respected Dr. Pooja Mittal (Research Guide and Second Author). Author's Qualification is M.Phil. (CS), MCA. She had attended many National and International Conferences including Springer and IEEE and also published many research papers. She can be contacted at email: poonam.mehta20@gmail.com.



Dr. Pooja Mittal    obtained her Ph.D. degree from Maharshi Dayanand University. Her area of research and specialization include Data Mining, Data Warehousing, and Computer Science. She had published more than 50 research papers in renowned International and National Journals and attended more than 30 Conferences. Currently she is working as Assistant Professor in the Department of Computer Science & Applications, Maharshi Dayanand University, Rohtak (Haryana). She can be contacted at email: mpoojamdu@gmail.com.